A Framework for combining Rules and Geo-ontologies

Philip D. Smart[1], Alia I. Abdelmoty[1], Baher A. El-Geresy[2], and Christopher B. Jones[1]

[1] Cardiff School of Computer Science,
Cardiff University, Wales, UK
[2] School of Computing,
University of Glamorgan, Wales, UK

Abstract. Geo-ontologies have a key role to play in the development of the geospatial-semantic web, with regard to facilitating the search for geographical information and resources. They normally hold large amounts of geographic information and undergo a continuous process of revision and update. Hence, means of ensuring their integrity are crucial and needed to allow them to serve their purpose. This paper proposes the use of qualitative spatial reasoning as a tool to support the development of a geo-ontology management system. A new framework for the representation of and reasoning over geo-ontologies is presented using the web ontology language (OWL) and its associated reasoning tools. Spatial reasoning and integrity rules are represented using a spatial rule engine extension to the reasoning tools associated with OWL. The components of the framework are described and the implementation of the spatial reasoning engine is presented. This work is a step towards the realisation of a complete geo-ontology management system for the semantic web.

1 Introduction

Retrieval of geographically-referenced information on the Internet is now a common activity. A large number of documents stored and retrieved on the web include references to geographic information, typically by means of place names. Also, the web is increasingly being seen as a medium for the storage and exchange of geographic data sets in the form of maps. The geospatial-semantic web (GeoWeb) is being developed to address the need for access to current and accurate geo-information [6]. The potential applications of the GeoWeb are numerous, ranging from specialised application domains for storing and analysing geo-information to more common applications by casual users for querying and visualising geo-data, e.g. finding locations of services, descriptions of routes, etc.

At the heart of the GeoWeb are geographic ontologies or geo-ontologies. These are models of terminology and structure of geographic space as well as records of entities in this space. An example of such an ontology has been proposed recently in the SPIRIT project [14] and was shown to play a central role in

the development of a geographical search engine. Building geo-ontologies involves a continuous process of update to the originally modeled data to reflect change over time as well as to allow for ontology expansion by integrating new data sets, possibly from different sources. One of the main challenges in this process is finding means of ensuring the integrity of the geo-ontology and maintaining its consistency upon further evolution. Developing methods for the management of the spatial integrity of geo-ontologies will contribute towards the development of reliable geographical search engines and to the success of the GeoWeb in general.

In this paper we propose a new framework for the management of geo-ontologies for the purpose of geo-information retrieval. In particular, we build upon and utilise research results in the area of qualitative spatial reasoning (QSR). Composition tables for different types of qualitative spatial relations are used to derive general rules that govern the structure of the geographic entities and their interaction in space. A spatial integrity rule language has been developed, as an extension to OWL , for the expression of these rules. OWL and the popular semantic web reasoning engine Jena, are used for the representation and reasoning over the geo-ontology. This paper describes the new framework proposed and the implementation of the spatial reasoning engine. The presentation is limited only to the main distinguishing characteristics and extensions realised. The design of the language, its syntax and semantics are outside the scope of this paper.

Section 2 introduces the need of rules for supporting the representation and management of geo-ontologies and summarises the requirements of a spatial rule language for geo-ontologies. An overview of the new proposed framework proposed is given in section 3. Section 4 describes in some detail the implementation of the spatial rule engine, followed by section 5 which shows some examples to demonstrate the developed system. Conclusions and a view of ongoing research work is presented in section 6.

2 Rules for Geo-ontologies

Work is ongoing on the development of geo-ontologies to capture the conceptualisations of geographic domains and to facilitate the reuse and sharing of the geo-referenced information on the web. Several examples of geo-ontology developments have recently been proposed [9, 5].
The following are some of the particular distinguishing characteristics of geo-ontologies of interest to this work.

1. Geo-ontologies are normally associated with large instance bases (or A-boxes). A geo-object can have one or multiple spatial representations to define its location in space. For example, a city may be associated with a polygon object made up of hundreds of points representing its boundary, a simplified bounding box approximating its shape, as well as a point representing its centre. Large instance bases and multiple spatial representation lead to large ontology files and associated overheads.

2. Much of the semantics in geo-ontologies are implicit and evident only at the instance level. For example, different types of spatial relationships exist between every object and all other objects in space; an object may be inside, north-of, near to, larger than another object, etc. Some of those relationships may be captured on the concept level but most others are implicit, evident only by visual interpretation and geometric computation. Explicit representation of such relationships is not practically possible and means for their automatic extraction are needed.
3. Maintaining the logical as well as spatial integrity of geo-ontologies is crucial for maintaining their soundness and viability. Spatial integrity is different, and perhaps more complex, than logical integrity. Logical consistency does not automatically enforce spatial consistency. For example, a part-of semantic relationship between two geo-objects does not imply directly the correct relationships between the objects' spatial representations. The boundary of the child object might intersect with the parent or the area of the child might be larger than the parent, etc. If a third object exists that is located completely outside the parent object, then it is an error to insert a fact that this object intersects the child.

An understanding of the rules that govern space and spatial relationships is needed for the specification of spatial integrity rules to maintain the consistency of geo-ontologies. The problem is also evident when integrated utilisation of multiple geo-ontologies is considered, where processes such as comparison and merging assume the consistency of the candidate ontologies.

In [1], we reviewed the potential and limitations of OWL for representing geo-ontologies, the challenges indicated above can't be addressed directly using OWL. Recently, rule languages have been proposed that complement and enhance the expressiveness of standard ontology languages. Rule expression over geo-ontologies is needed for the representation of the following types of rules:

- Spatial reasoning rules for the deduction of implicit geo-semantics.
- Spatial integrity rules for representing different type of spatial integrity constraints to maintain the consistency of geo-ontologies.

In the rest of this section, spatial reasoning techniques are reviewed that allow for the identification and expression of both of the above rule types.

2.1 Qualitative Spatial Reasoning Tools

In this section, we demonstrate examples of the use and adaptation of some types of spatial reasoning techniques and the derivation of spatial rules, that are used as a basis for the spatial reasoning engine in the framework described later in the paper. A possible classification of the types of spatial rules is as follows.

- Rules representing constraints over object properties in space, in particular, spatial properties of dimension, shape and size. Examples of these types of rules include the fact that a polygon must have at least three different

points and that a polygon must be closed, etc. These types of constraints are normally used in spatial databases and GIS.

– Rules for reasoning over spatial relationships between objects in space. For example, the fact that an object A is located inside another object B and that B is inside object C, implies that object A is also inside C. It also implies that C is larger than A and B. This is an example of qualitative spatial reasoning (QSR). Here, we utilise the results of the large body of research in this field, where automated methods have been proposed for the derivation of spatial composition tables for different types of spatial objects and relationships. Table 1 shows part of a composition table for topological relations between two simple regions.

	$d(y,z)$	$m(y,z)$ 	$i(y,z)$ 	$ct(y,z)$ 	$o(y,z)$ 
$d(x,y)$	all	$d \vee m \vee i \vee o$	$d \vee m \vee i \vee o$	d	$d \vee m \vee i \vee o$
$m(x,y)$	$d \vee m \vee ct \vee o$	$d \vee m \vee i \vee ct \vee o$	$i \vee o$ o	d	$d \vee m \vee i \vee o$
$i(x,y)$	d	d	i	all	$d \vee m \vee i \vee o$

Table 1. Composition table for the set of base topological relations between simple regions.

Entries in the composition tables can be encoded into rules that can be used as deduction rules for the automatic derivation of implicit spatial relationships, as well as constraints for enforcing the integrity of the spatial data sets. These constraints are the building blocks of the proposed spatial reasoning engine as described later in the paper.

When reasoning over networks of spatial objects as with a typical geo-ontology, QSR becomes a more general constraint satisfaction problem. A path consistency algorithm was proposed earlier to address this problem [18, 15]. The main function of this algorithm is denoted *REVISE* which deduces the consistency of region triples {A,B,C} by performing the following operation. [3]

$$A_rC = A_rC \cap (A_rB \otimes B_rC)$$

The equation validates whether the known or explicitly specified relationship(s) between A and C, contradicts the relationship(s) that may be derived between the same two objects, using the composition of their relationships with

[3] where $\otimes$ represents the composition of spatial relationships

other objects in the scene (B in this case). The function was first used in the temporal domain by Allen [2] in his work on interval calculi. The implementation of the algorithm relies on the existence of pre-specified spatial composition tables. If the composition returns an empty set, the scene is inconsistent, otherwise other regions are selected and the process of spatial composition and intersection is repeated for the rest of the objects in the scene.

2.2 Requirements for a Spatial Rule Language for Geo-ontologies

From the above section a list of requirements can be drawn for the design of a spatial rule language. Standard characteristics of a general rule language, designed to work with ontology languages, e.g. SWRL [13] or RuleML [22], are assumed. The following list are desirable additional characteristics for rule languages in the spatial domain. The specification of the language design and the language semantics is out of the scope of the current paper.

- Assumes a standard spatial data model (conforming with OGC or ISO spatial models). Predicates in the language will represent different types of geo-features, their associated geometric representations as well as different types of spatial operators and relationships.
- Can represent absolute spatial constraints on geographic features.
- Can represent relative spatial constraints between geographic features, including, topological, directional and proximity.
- Allows for external calls to geometric processing functions for the evaluation of pre-specified types of spatial relationships. As explained in the above section, only some spatial relationships can be stored a priori in the fact base. The application of spatial reasoning rules will occasionally require the evaluation of some of the implicit relationships using computational geometry algorithms supported by spatial database systems or GIS.
- Allows for the expression of rule exceptions. This characteristic is particularly useful for the expression of application specific rules, where in some cases exceptions to general spacial rules are required. See section 4.4 for an example.

The language should also have a formal logical underpinning, clear semantics and be serializable into a RuleML representation in order to interface with existing semantic web technologies.

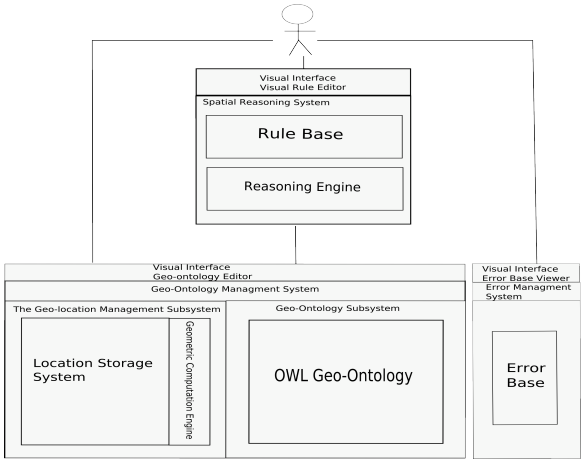
3 Geo-Ontology Management System Framework

The main objectives for the geo-ontology management system proposed here are to support the representation and storage of geo-ontologies and to allow for the expression and realisation of spatial integrity maintenance and deduction rules over the geo-ontologies.

Hence, the new framework proposed consists of three main component subsystems that together demonstrates an architecture for a system that allows for

the spatial integrity maintenance management of geo-ontologies. These systems are: 1. the geo-ontology system and associated geo-location storage system, 2. the spatial rules management and the spatial inference engine, and 3. the error management system. The framework is shown in Figure 1. The system has been implemented using OWL , the Jena toolkit and Oracle Spatial.

Fig. 1. A new framework for representing and maintaining geo-ontologies



3.1 The Geo-ontology Management System

Given the limitations of current web ontology languages for representing geographic features and their geometry, a dual model of representation will be used. The geographic concepts and features in the geo-ontology will be represented using OWL, while the spatial representations of the geo-features will be modeled using an external geometric processing or spatial database system. Such a dichotomy of representation does not affect the validity of the overall framework and is proposed as a practical solution to overcome the limitations of the current semantic web tools. The same spatial integrity maintenance framework will operate on geo-ontologies completely represented in OWL.

Geo-Ontology Subsystem The geo-ontology's spatial data model conforms to the OGC abstract feature specification . The model also assumes a predefined set of qualitative spatial relationship properties including topological, directional as well as relative proximity and size relationships.

The Geo-location Management Subsystem The types of geometric computation operations, such as distance or area, required to operate over locational information are not supported using OWL's schema or functions. Hence, the representation and management of the absolute locational information are delegated to an external geometric processor or a spatial database system, referred to as the Location Storage System (LSS) in Figure 1. Such systems have efficient spatial indexing techniques and optimised geometric processing capabilities.

A unique reference between features in the ontology and their corresponding locational information represented in the LSS is maintained. As URIs provide, what we will assume to be, a unique name to the features of the ontology, they will be mapped directly into the LSS as primary keys. Note that a single interface is used in the framework to both the OWL geo-ontology and the LSS that together form the complete geo-ontology used by the rest of the framework components.

3.2 The Error Management System

Errors mined from the geo-ontology by integrity rules are stored in a separate error ontology. Building an error ontology is interesting as it provides opportunities for reasoning over errors and would, for example, give insight to the types of integrity problems, their frequency and guide the error management process.

3.3 The Spatial Reasoning System

The spatial reasoning system (SRS) is at the heart of the framework. The system provides the functionality to represent spatial rules in their native format. the SRS is implemented in Java and sits alongside the Jena toolkit. Jena is an open source Java-based semantic web toolkit. Jena provides an API to access, manipulate and reason with RDF and OWL ontologies. Jena's rule reasoning engine uses a Rete-based forward production rule engine [24] , along with an XSB [23] based backward chaining logic programming engine.

Jena's reasoning subsystem is limited when it comes to the authoring of rule sets. The SRS implements a complete rule authoring system to construct, store, modify and visualise a spatial rule set. As the spatial rule set is syntactically and semantically different from a Jena rule set, SRS translates spatial rule sets into a format compliant with Jena for the purpose of execution. It also defines extensions to the rule engine in Jena to realise the full expressive extent of spatial reasoning rules. A more detailed overview of this system is presented below.

4 The Spatial Reasoning Engine

As mentioned, our proposed spatial rule language and reasoning engine have been implemented using the Jena toolkit. In what follows we describe the specific extensions to the toolkit needed to address the requirements of the rule language identified earlier. A more exhaustive treatment of the extensions described and their logical underpinnings are the subject of another report.

4.1 Interleaved Execution Extension

Typically, all rule body antecedents are matched from existing stored facts (that being facts derived by rules or explicitly represented). By interleaving forward and backward reasoning modes, facts can be derived, or proven, on the fly by a set of one or more backward rules. This is useful is minimising storage overheads. Consider for example the following rule:

$$[Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge Inside(?x?c) \wedge Inside(?c?y) \rightarrow Inside(?x?y)]$$

The conclusion of `Inside(?x ?c)` would only be inferred if both the atoms `Inside(?x ?c)` and `Inside(?c ?y)` can be satisfied. These atoms are either satisfied by facts directly stored in the ontology (explicit), or inferred using QSR rules (implicit)[4], or as a last resort satisfied by a rule that calls an external geo-computation engine.

For example, the following is a subset of QSR rules used to derive the inside relationship between two regions. The fifth rule is a call to external geo-computation (*exInside* predicate).

$$Inside(?x\ ?y) \leftarrow Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge Inside(?x\ ?c) \wedge Equal(?c\ ?y)$$
$$Inside(?x\ ?y) \leftarrow Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge Inside(?x\ ?c) \wedge Inside(?c\ ?y)$$
$$Inside(?x\ ?y) \leftarrow Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge Inside(?x\ ?c) \wedge CoveredBy(?c\ ?y)$$
$$Inside(?x\ ?y) \leftarrow Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge CoveredBy(?x\ ?c) \wedge Inside(?c\ ?y)$$
$$Inside(?x\ ?y) \leftarrow Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge exInside(?c\ ?y)$$

Besides using builtins to evaluate spatial relationships, the engine supports a standardised set of predefined spatial builtins, such as for example, simple arithmetic and comparison operators that are evaluated using the external geometric processor. The restriction that variables must be bound applies to any external call, i.e. all variable must be bound and the call surmounts to a test of truth (returning either True or False). Interleaved logic programs have been implemented to varying degrees in Algernon, the M.4 system, MIKE, ECLIPSE and Harlequin.

Interleaved Implementation Jena, more specifically Rete, does not inherently provide a means to call a backward rule during the course of antecedent pattern matching. To support this feature a backward call is added as a builtin (such a predicate is henceforth denoted a reserved spatial relation predicate). That is, the reserved spatial relationship predicates are not represented as triple patterns, but are added to the engine as builtins. The builtins are coded in Java and are registered with Jena's forward engine.

Once the builtin is called, the backward rule engine is initialised over the current set of intentional and extensional triples. For efficiency of retrieval using the external geometric processor, calls to backward rules must only contain ground variables. Thus backward rules only evaluate one relationship between

[4] Spatial reasoning rules are defined using composition tables as described earlier

two features at a time, and as such will either return true or false. For example, consider the following forward rule.

$$Region(?x) \wedge Region(?y) \wedge Region(?c) \wedge Inside(?x\ ?c) \rightarrow \cdots$$

`Inside(?A ?B)` represents a query to the backward QSR rule set. The nature of builtins in Jena, and the two class predicates, ensures that the variables ?A and ?B will be bound before the backward query is executed. Hence, `Inside(?A ?B)` will return either true or false, based on whether that relationship exists in the ontology, can be inferred, or whether it can be determined from the geometry.

Jena's Backward Engine: The reserved spatial relation predicates are executed in Jena's integrated XSB backward engine. XSB is based on a modified version of SLD resolution, namely SLG resolution. The following features of XSB are of particular interest to this work.

1. SLG tabling allows the transitive closure of a property to be computed without entering an infinite loop - as would be the case with SLD resolution. This is very useful for example when computing the containment hierarchy of geofeatures.
2. SLG's left to right top to bottom procedural reading (first in first out, also a feature of SLD resolution). The efficiency of the system is heavily affected by this. Rules with shallow inference chains can be evaluated before rules with deeper inference chains, lastly followed by rules with external calls to spatial relationship computation.

The order with which rules in the backward rule set are executed is explicitly defined by the rule execution metadata tag. In Jena, the order of rule evaluation is determined by the order in which the rules are encoded in the rule string sent to the backward reasoning engine. Therefore during conversion from the rule system to Jena, the rule string is constructed in the order that is represented by the backward meta tags.

4.2 Integrity Rules

The bodies of integrity and deduction rules are identical in both specification and functionality. An integrity rule differs from a deduction rule in the use of its head atom. That is, an integrity rule does not assert new information into the ontology[5], instead it asserts errors into an error ontology (thus permitting the storage of errors). Positive and negative errors can be concluded, as explained below.

[5] As is common in logic programming literature, a rule without head is referred to as an integrity rule

Default Integrity Rules As indicated in the requirements earlier, it is desirable for the rule language to represent default integrity rules and their exceptions. This is a form of default reasoning [21, 17, 20]. That is, a default rule is assumed true until its contrary can be proved. For example:

$$body \rightarrow error(X)$$
$$body2 \rightarrow \neg error(X)$$

where X is a variable. The error of the first rule is assumed until there is enough evidence to support body2, and the error (where both errors have the same variable substitution for X) is refuted.

A large body of research in the area of prioritised default reasoning has studied this problem. Courteous logic [10] is a popular type of prioritised default reasoning which is expressive enough to capture our integrity requirements. Courteous logic provides us with a natural and intuitive way to provide rule priority to capture the requirement of rule defaults and rule exceptions.

Example: The following is an example of a spatial integrity constraint with both a default rule and an exception to that rule.

$$Road(?x) \wedge River(?y) \wedge Crosses(?x?y) \rightarrow$$
$$error(roadRiverCrossError\ ?xCrosses\ ?y\ doNotCross\ riverRCross) \quad (1)$$
$$Road(A40) \wedge River(Taff) \wedge Crosses(A40\ Taff) \rightarrow$$
$$notError(roadRiverCrossError\ A40\ Crosses\ Taff$$
$$roadsRiversDoCross\ riverRCrossException) \quad (2)$$

Rule (1) is the default rule and (2) its exception. Intuitively, The ground instantiation of the first rule which substitutes variables ?x and ?y for A40 and Taff respectively is overridden by the second rule.

An often used first step to the implementation of Courteous Logic in current reasoning engines is through the use of a courteous compiler. A courteous compiler compiles away the expressive Courteous logic extensions, leaving a semantically equivalent ordinary logic program [11], which can then be implemented in common logic program reasoning engines such as PROLOG. Here, we adapt such an implementation by placing some expressive restrictions on the courteous logic component, thus removing the need for a courteous compiler. Instead, we employ a simple algorithm denoted, the Prioritized Conflict Handling Engine ($PCHEng$), to perform a post processing cleanup. The following are the expressive restrictions to the full Generalised Courteous Logic as described in [12].

1. As with a basic courteous logic program we permit only the classical mutex.
2. Classical negation is restricted to integrity rule head atoms only, i.e. to infer $error$ and its negation $\neg error$. Negation as failure is completely removed.

We have, however, in part extended the Courteous Logic specification. That is, we have supplemented the rule label with additional types of rule meta data or tags, which can be used to infer priorities amongst integrity rules.

With the above restrictions, a rule is definite. That is, it does not contain negation as failure and the limited form of classical negation can be dealt with by the $PCHEng$ post processing transform. Our simplified version of Courteous Logic will be henceforth denoted CLP^{-}. The advantages of using the CLP^{-} approach are two fold. Firstly, we need not deal with the rather complex semantics of a logic program that contains negation as failure (stable models [8] etc). Secondly, it allows the dynamic generation of rule priorities based on reasoning over rule meta tags and inferring *Overrides* predicates.

There are a number of implementations of default or defeasible reasoning, namely DR-Prolog [3], DR-Device [4], DELORES -a Defeasible Logic Reasoning System [16]. Both DR-Prolog and DR-Device handle non-monotonic rules over RDFS ontologies. DR-Prolog is implemented by transforming information into PROLOG, and DR-Device works by transforming information into JESS. All lack procedural attachment. Defeasible reasoning with procedural attachments is supported by the SweetRules project [19]. SweetRules supports Situated Courteous Logic, that is, Courteous Logic with cleanly formalised procedural attachments.

***CLP*$^{-}$ Implementation** The implementation of CLP^{-} can be divided into two stages:

Stage 1: Jena's implementation of Rete [7] for forward inferencing does not support strong negation ($\neg$, more akin to classical negation) - indeed Rete in general lacks support for classical negation. Therefore the first step involves the removal of all appearances of classical negation. This is an easy step and is a common way of adding a limited form of classical negation in ordinary or definite logic programs [11]. The step involves: for each error predicate *error*, each appearance of $\neg error$ is replaced by an appearance of a new predicate notError; and a new explicit mutex between *error* and *notError* is introduced - or assumed (as we only deal with the classical mutex).

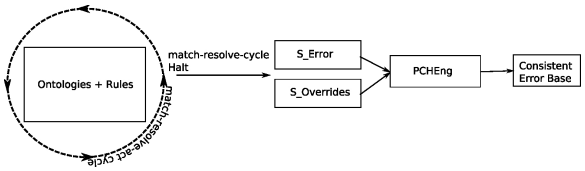
Stage 2: At the end of the inferencing stage, when Rete's match-resolve-act cycle has halted, a potentially inconsistent error base may result. That is, for all predicates in the error base, some may be negatively and positively represented. The error base is then fed into the Prioritised Conflict Handling Engine ($PCHEng$) along with the *Overrides* sub program, see figure 2. Stage 2 is performed by the following algorithm.

Algorithm: In overview, the PCH engine removes two conflicting error predicates by checking for a relevant *Overrides* predicate with which to resolve the conflict. As is the norm with a CLP, if an *Overrides* can't be found, then both positive and negative versions of the error are removed - treated skeptically.

4.3 Metadata

Rule metadata serves two purposes in our system. The first is as a form of reflection to derive *overrides* facts used by the CLP^{-} component. Secondly, to

Fig. 2. *PCHEng* Information Flow



facilitate the visualisation and authoring of large spatial rule sets. Syntactically rule meta data is represented during the rules preamble. For example:

```
[<meta-data> : BODY -> HEAD]
```

The syntactical representation of rules separates the appearance of the rules meta data from the rules logic. However, in order to reason over rule meta data during the reasoning process, the tags must be translated to syntactically reserved, variable free predicates.

Our rule language supports the following spatial meta tags.

```
forward_meta_data = "<" rule_Name "> "<" rule_Level "> "<" rule_Type ">
                    "<" rule_Class "> "<" spatial_Rule_Group ">
backward_meta_data = forward_meta_data "<" backward_Rule_Group ">
                    "<" backward_Rule_Order ">
```

Overrides predicate inference example: Meta level reasoning can be used to infer a general prioritisation between level 0 and level 1 rules, for example:

$$[ruleLevel(?A\ 0)\ AND\ ruleLevel(?B\ 1) \rightarrow overrides(?A\ ?B)]$$

As a result, all rules that have a rule level of 0 will override rules having a level of 1 - providing they have conflicting error predicates.

4.4 Example of Default Reasoning

In this example consider the domain specific knowledge that roads and rivers do not cross in the general case. There are, however, exceptions to this rule. A road may pass through a river where the river is shallow enough (a forge). Without modeling forges directly, integrity rules can be used to capture this situation using a form of default reasoning.

Consider for example, that roads A40, A50 and the rivers Taff and Tywn are instantiated into a geo-ontology, and that the spatial relationships {A40 crosses Taff} and {A50 crosses Tywn } are also added. The following rule is then added to the rule base as a default.

[< *label* > *riverRoadCross* < */label* >< *ruleLevel* > 0 < */ruleLevel* >< *ruleGroup* > *Topo* − *Semantic* < */ruleGroup* >< *ruleType* > 0 < */ruleType* >< *ruleClass* > 1 < */ruleClass* >: Road(?x) AND River(?y) AND Cross(?x ?y) → error(roadRiverCrossError ?x Crosses ?y roads_rivers_do_not_cross riverRoadCross)]

Algorithm 1 PCHEng

```
Let S = array of all error individuals in the error ontology
Let P = array of 2-tuple records representing conflicting errors (error, error) - conflict
set
Let Ov = array of all overrides predicates
for (i =0; i < sizeof(S);i++) do
   for (int j=0; j < sizeof(S);j++) do
      if (i ≠ j) then
         if (s[i] complementof s[j]) then
            add s[i] and s[j] to P
         end if
      end if
   end for
end for
for (int i=0; i < sizeof(P); i++) do
   Let found = FALSE
   for (int j =0; j < sizeOf(Ov); j++) do
      if (Ov[j] represents priority over P[i]) then
         Remove defeated error triple
         Set found = true
      end if
   end for
   if (found == false) then
      remove both error triples
   end if
end for
```

With only the default rule in the rule base, both the A40 and A50 are added to the error base as shown in figure 3. I.e. roads should not cross rivers.

A further rule is asserted into the knowledge base that contradicts the default rule by specifying the negation of the error that occurs when the individuals assigned to the variables ?x and ?y are A40 and Taff respectively.

[< *label* > *riverRoadCrossException* < */label* >< *ruleLevel* > 1 < */ruleLevel* >< *ruleGroup* > *Topo − Semantic* < */ruleGroup* >< *ruleType* > 0 < */ruleType* >< *ruleClass* > 0 < */ruleClass* >: Road(A40) AND River(Taff) AND Cross(A40 Taff) → notError(roadRiverCrossError http://phils.sorl.ont/A40 Crosses http://phils.sorl.ont/Taff roads_rivers_do_cross riverRoadCrossException)]

A rule is used to represent the fact that all level 1 rules override all level 0 rules. As a result, because the exception being at a lower level than the default, the error for that instance is eliminated.

[< *label* > *overrides* < */label* >:ruleLevel(?x 0) AND ruleLevel(?y 1) → overrides(?x ?y)]

With both the rule exception and the overrides rule now added to the rule system, only the error between the A50 and Twyn is detected as shown in figure 4.

Fig. 3. Error Base Without Exception Rule(s)

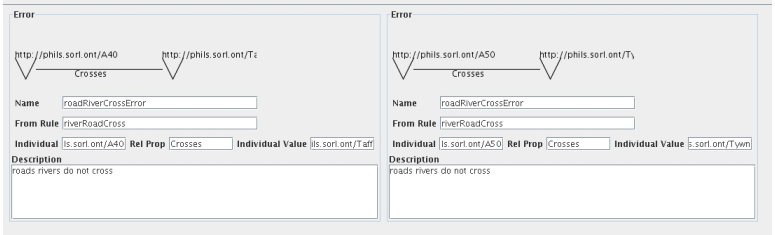
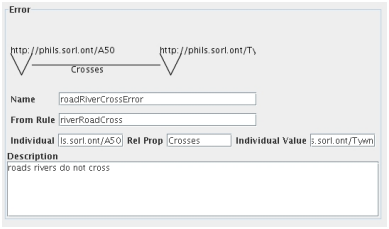


Fig. 4. Error Base After Exception Rule(s) Added



5 Conclusions

A new framework for the representation and management of geo-ontologies is proposed. Rules for geo-ontologies were shown to serve two primary purposes, namely, deduction of implicit spatial semantics and expressions of spatial integrity constraints. Requirements for a spatial rule language for geo-ontologies are identified and are used as a base for the design and development of a spatial reasoning engine. Particular extensions to support the desired requirements to the Jena toolkit are described and some examples are given to demonstrate the developed system. The system developed implements a new spatial rule language for geo-ontologies and has been tested and evaluated using synthetic and realistic geo-ontologies, partly within the scope of the EU SPIRIT project. The design of the language and details of the evaluation experiments are out of the scope of this paper.

References

1. Abdelmoty, A. I., Smart, P. D., Jones, C. B., Fu, G., and Finch, D. *A critical evaluation of ontology languages for geographic information retrieval on the Internet.* Journal of Visual Languages & Computing. August 2005, pp. 331–358.

2. Allen, J. F. Maintaining knowledge about temporal intervals. Tech. rep., University of Rochester, Department of Computer Science, 1981.
3. Antoniou, G., and Bikakis, A. DR-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Trans. Knowl. Data Eng 19*, 2 (2007), 233–245.
4. Bassiliades, N., Antoniou, G., and Vlahavas, I. P. DR-DEVICE: A defeasible logic system for the semantic web. In *PPSWR* (2004), H. J. Ohlbach and S. Schaffert, Eds., vol. 3208 of *Lecture Notes in Computer Science*, Springer, pp. 134–148.
5. Bernard, L., Einspanier, U., and Haubrock, S. Ontology-based discovery and retrieval of geographic information in spatial data infrastructures. In *Geotechnologien Science Report No. 4* (2004). http://www.delphi-imm.de/meanings/index_eng.html.
6. Egenhofer, M. J. Toward the semantic geospatial web. In *Proceedings of the tenth ACM international symposium on Advances in geographic information systems* (2002), ACM Press, pp. 1–4.
7. Forgy, C. Rete: A fast algorithm for the many patterns/many objects match problem. *Artif. Intell 19*, 1 (1982), 17–37.
8. Gelfond, M., and Lifschitz, V. The stable model semantics for logic programming. 1070–1080.
9. Goodwin, J. Experiences of using owl at the ordnance survey. 1–11. http://www.mindswap.org/2005/OWLWorkshop/.
10. Grosof, B. N. Prioritized conflict handling for logic programs. 197–211.
11. Grosof, B. N. Compiling prioritized default rules into ordinary logic programs, June 23 1999.
12. Grosof, B. N. DIPLOMAT: Compiling prioritized default rules into ordinary logic programs, for E-commerce applications. In *AAAI/IAAI*. 1999, pp. 912–913.
13. Horrocks, I., Patel-Schneider, P. F., Tabet, H. B. S., Grosof, B., and Dean, M. Swrl: A semantic web rule language combining owl and ruleml. Internet Report, May 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
14. Jones, C., Abdelmoty, A., and Fu, G. Maintaining ontologies for geographical information retrieval on the web, 2003.
15. Mackworth, A. Consistency in networks of constraints. *Artificial Intelligence 8* (1977).
16. Maher, M., and Miller, T. Delores a defeasible logic reasoning system. internet, 2007. http://www.nothingisreal.com/dfki/delores/.
17. McCarthy, J. Circumscription, a form of non-monotonic reasoning. *Artificial Intelligence 13* (1990), 27–39.
18. Montanari, U. Networks of constraints: Fundamental properties and application to picture processing. In *Information Science* (1974), vol. 7.
19. Neogy, C. C., (UMBC), S. G., Grosof, B., Dean, M., and Tabet, S. Sweetrules. Internet, 2006. http://sweetrules.projects.semwebcentral.org/.
20. Nute, D. General Defeasible Logic. Tech. Rep. forthcoming, University of Georgia, 1989.
21. Reiter, R. A logic for default reasoning. *AI 13* (1980), 81–132.
22. RuleML. Ruleml web site. Internet, 2006. http://www.ruleml.org/.
23. Sagonas, K., Swift, T., and Warren, D. S. Xsb: An overview of its use and implementation. Tech. rep., Nov. 02 1993.
24. Schneier, B. The rete matching algorithm. *AI Expert 7*, 12 (Dec. 1992), 24–29.